MESA: An Interactive Modeling and Simulation Environment for Intelligent Systems Automation

Leonard Charest, Jr., Nicolas Rouquette, Richard Doyle

Artificial Intelligence Group
Jet Propulsion laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
Mail Stop 525-3660
(818) 306-6143
{charest,rouquett}@aig.jpl.nasa.gov rdoyle@isd.jpl.nasa.gov

Charles Robertson, Tim Hill

McDonnell Douglas Aerospace
Space Station Division
13100 Space Center Boulevard
Houston, TX 77059
Mail Stop MDC1-625GB
(713)283-1080
{robertso, hill}@fdm.mdc.com

E. Jay Wyatt

JPL, wyatt@aig.jpl.nasa.gov

1 introduction

This report describes MESA[1], a software environment for creating applications that automate NASA mission operations. MESA enables intelligent automation by utilizing model-based reasoning techniques developed in the field of Artificial Intelligence. Model-based reasoning techniques are realized in Mesa through native support of causal modeling and discrete event simulation,

This report will focus on MESA as a tool for model development and model-based reasoning in problem domains specific to NASA. This sect ion continues with a discussion of the background and objectives that motivated the development of MESA. Section 2 describes the concepts and terminology of causal modeling as implemented in MESA. Section 3 uses an example modeling session to give a conceptual overview of MESA's architecture and user interaction paradigm. Finally, Section 4 summarizes the status of our ongoing experimentation with MESA in NASA problem domains.

[1]MESA stands for Modeling Environment for Systems Automation.

1.1 Background

Within NASA, there are great pressures to reduce the cost of mission development and operations while maintaining reliability and effectiveness. One approach that is increasingly desired and often necessary is to automate monitoring and diagnosis of spaceborne systems through the deployment of software applications which utilize a computer-based model of these systems. However, there are several bottlenecks associated with ongoing automation efforts. For example, the overhead of model development can be prohibitively large. Also, model design is often bound to a specific mission application, thereby inhibiting model reuse.

To lower the development and deployment costs of a given mission, as well as guarantee a level of correctness, the technology base for monitoring and diagnosis should be generic *and* easily customizable for a specific mission. These constraints are satisfied by a modeling environment that provides a generic model representation scheme as the foundation for a suite of robust tools for monitoring and diagnosis. MESA was built on the premise that model-based reasoning techniques for monitoring and diagnosis can use a generic model representation as a stepping stone to focus development efforts on reusable models and low-cost custom applications.

models in a component-centered, object-oriented fashion [5, 1]. Together, EDSE and EDSEL form a robust tool for evaluating applications of model-based reasoning. EDSE features a user-programmable simulation protocol which enables a wide range of application-specific model-based reasoning strategies. EDSEL employs the concepts of data abstraction and inheritance from the computer programming discipline to promote the reuse of model constituent definitions.

However, this tool is difficult to use for both developers and domain experts since it does not provide a complete modeling environment. Specifically, EDSE is not interactive; the user is required to have knowledge of the Lisp environment in which EDSE is hosted; EDSE does not support graphical presentation of simulation results; and finally, the tool offers no support for incremental model development,

Our objectives for synthesizing MESA are twofold: First, we desire an interactive tool to assist our own efforts in developing and verifying causal models and in experimenting with the models through simulation and playback of historical data files. Such a tool should address the shortcomings of stand-alone EDSE described above. Second, we wish to provide our domain experts and process engineers with a knowledge acquisition tool to help them express causal relationships within and amongst the components

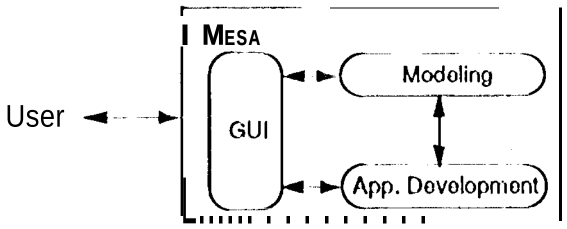


Figure 1: Conceptual view of MESA.

of a physical system of interest.

1.3 Architecture

MESA achieves our objectives by integrating EDSE and EDSEL with a graphical user interface (GUI) that encourages model development and model-based reasoning efforts to be interleaved. The environment makes modeling and model-based reasoning tools available at all phases of application development. in this manner, models and applications evolve as a consequence of incremental changes to an initial design.

Conceptually, MESA comprises three distinct but cooperative pieces as shown in Figure 1. EDSE and EDSEL (really its compiler) are essentially embedded subsystems of MESA.

As implemented, MESA consists of two computational processes that communicate asynchronously: Graphical model-building tools are provided by G2, a commercially available real-time expert system shell. The G2 process also comprises a translator that parses the contents of the G2 data structures that represent a causal model and writes the model to a file as EDSEL statements. A separate Lisp process executes EDSEL. Component and schematic libraries are stored in the file system of the host computer. All user interaction with MESA happens through the GUI provided by C;2, thereby unifying the MESA constituents under a single, consistent point-and-click interface,

2 Concepts in Causal Modeling

Causal models are an abstraction for describing the behavior of a system by representing the physical processes occurring within the system as discrete functions to be evaluated by the computer.

2.1 Causal Models and Simulation

A causal model characterizes a physics] system in terms of state variables and carnal influence relations among the variables. in the causal graph defined by the state variables and influence relations, changes in any variable may be propagated to other variables through the influence rclatires. Causal simulation is the process of tracking changes in variables and propagating them to other variables through influence relations, thereby producing a new set of changes [5]. Implicit in the notion of causality are the concepts of event and causal time: Events comprise changes in state variables due to a specific influence relation and with respect to a specific moment in time. Thereby, causal time moves forward due to delays in the propagation of changes in the causal model.

MESA represents causal models (at a low-level, via EDSE) in terms of the primitives de.scribed above. MESA adopts the following terminology: Causal time is expressed as a monotonically increasing sequence of integer-valued instants. Each state variable is denoted by a quantity of type integer, float or symbol. Each influence relation is described by a mechanism which encapsulates a set of input quantities, a set of output quantities, and transfer and delay functions. I bring simulation, the transfer function is evaluated with the mechanism input quantities to produce a value that will be prop agated to each of the mechanism output quantities. The delay function is simultaneously evaluated to produce an offset from the current simulation time. Value propagation takes place at the relative time returned by the delay function. Figure 2 visualizes mechanism evaluation. Simulation continues until a mm-specified time limit is reached or the model reaches

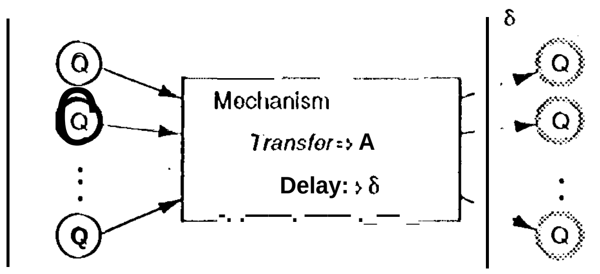


Figure 2: Causal simulation: A mechanism evaluated at time t_i will propagate its value at $t_{i+\delta}$.

quiescence.

2.2 Component-centered Modeling

At the user level, MESA adopts a component-centered approach to causal modeling. This modeling approach distinguishes the constituents of the system being modeled as components and the structural connections between them (Figure 3. A physical component is intuitively defined as a discrete element of a physical system, For example, in the EATCS evaporator loop schematic (Figure 4) we have identified CV1, CV2 and H1 as components. By analogy, a causal component is an entity that encapsulates the behavior of the corresponding physical component as a whole.

A connection is an abstraction which characterizes the interaction pathways between components. A connection within a physical system enables

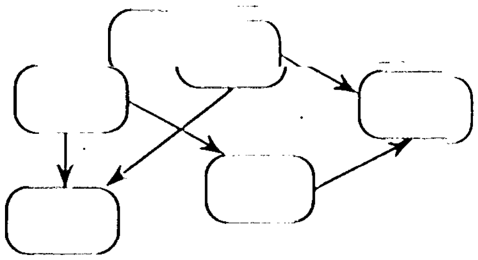


Figure 3: An abstract causal model. in the component-centered view, we reason about components (boxes) and the connections (arrows) between them.

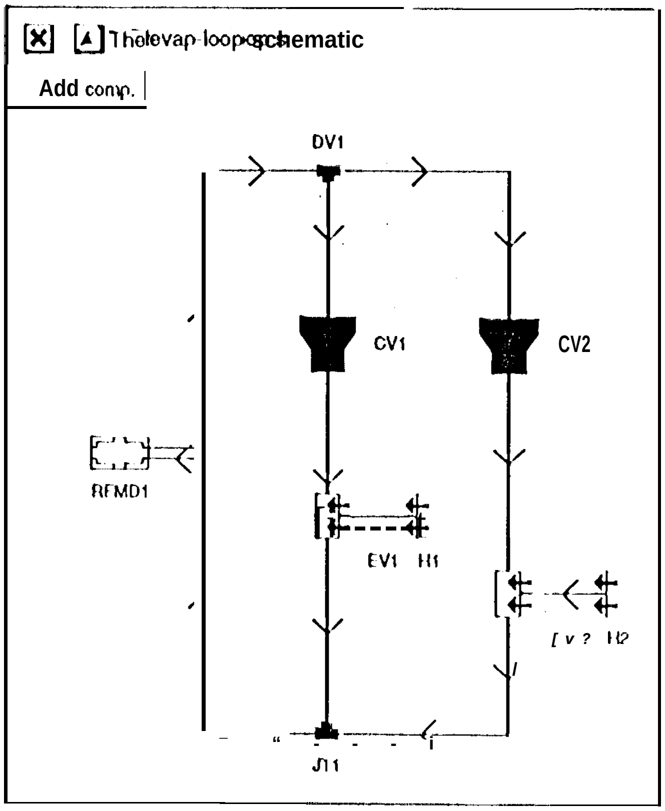


Figure 4: A simplified schematic of the EATCS evaporator loop.

material flow between components. Similarly, a causal connection allows quantity values to be propagated between components.

The component-centered approach to causal modeling engenders models which are isomorphic to a component-centered view of the physics] syst em being modeled. This resul t is advantageous since domain experts intuitively reason about systems from a component-centered perspective. A model formed by a configuration of causal components and connections (e.g., Figure 4) maps onto the schematic configuration of the corresponding physical system. MESA exploits this similarit y by presenting the outcome of model-based reasoning in a graphical format that is familiar to the domain experts (see Sect ion 43)0

2.3 Internal Representation of Components

Components may be regarded as a convenient abstraction for partitioning the causal graph into groups of semantically related quantities and mechanisms. Our experience with causal modeling has shown that it is useful and often necessary to subdivide the quantities into groups that represent the sources and sinks of the physical processes modeled by mechanisms.

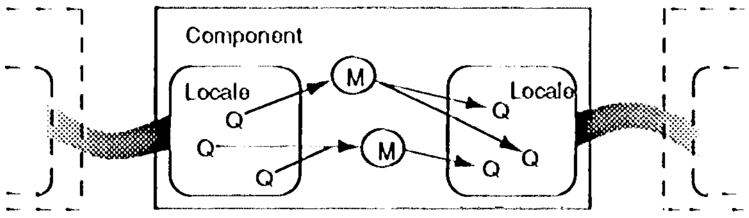


Figure 5: Internal representation and configuration of a component,

A locale is an abstraction for grouping quantities within a component.[2] For example, a pipe component could be modeled with separate locales for fluid flow at its inlet and outlet. in this fashion, locales serve as an isomorphism for locations of activity on a physical component.

Figure 5 visualizes the internal representation of components in MESA Note that mechanisms are not grouped with respect to a locale. Also, connections in MESA are realized at the locales of the component being connected.

3 User Processes and the GUI

This section details the mode and manner in which the user interacts with MESA. We begin with a brief description of the concepts in the GUI. The

[2]Full EDSEL syntax allows locales to be nested within one another, thereby creating a tree of locales. Each successive level in the tree denotes a smaller partitioning of the quantities on the component. Conversely, the component itself is considered to be an all-encompassing locale that forms the root of the locale tree.

remainder of this section uses a running example to illustrate the processes that inform model development Our example, a venturi component, is drawn from our diagnosis application for EATCS.

3.1 Concepts in the GUI

All of the MESA user interfaces are built using G2's GUIDE/UIL (Graphical User Interface Development Environment /User Interface Library) which is patterned after the Motif/X Windows paradigm The entry point into the MESA GUI is a control panel called the *Workbench* (Figure 6). Workbench contains several buttons which are used consistently across the GUI. A button labeled with the letter "X" dismisses the control panel on which it is located. A button labeled with a question mark "?" pops up a dialog of helpful information about the interface on which it is located. The buttons labeled with an arrow serve to navigate among the various control panels of the interface. Most control panels have a parent button (up-arrow) to switch to the parent control panel. Since the *Workbench* is MESA's top-level control panel, it has no parent button. The down-arrow buttons are used to pop up dialog boxes in which the user is prompted for information or a choice of actions.

One unique feature of the G2 GUI is the concept of workspace. A

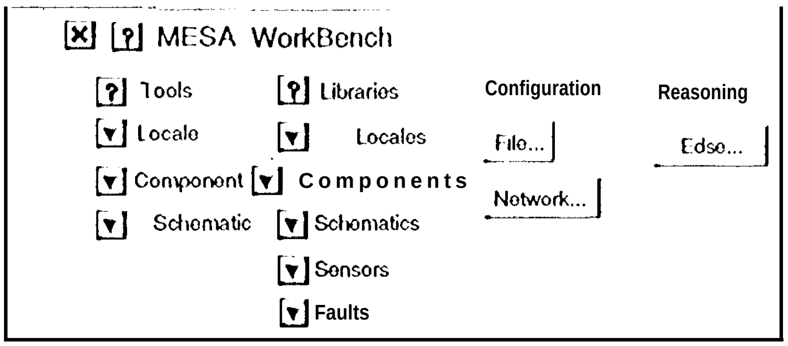


Figure 6: *Workbench* is the top-level control panel through which all modeling and nloclcl-based. reasoning tools can be accessed.

workspace is a rectangular area on the computer screen that serves to display and organize information in the form of icons. Each icon represents an object in the application hosted by C;2. icons also provide a direct-manipulation interface for accessing the attributes of each object. Subworkspaces may be stored on an icon, thereby allowing the user to create an hierarchy of workspaces.

3.2 User Processes

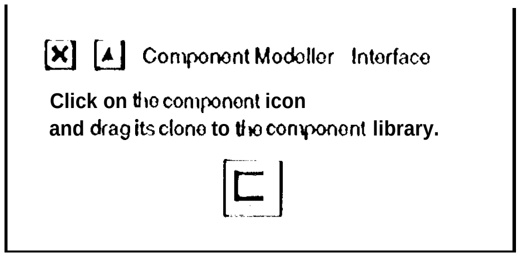
MESA addresses the development of models and applications with three user processes: mod el design, model assembl y and nlodcl-based reasoning.

3.2.1 Model Design

Model design is the process of ascertaining how physical components are to be modeled as causal components. First, the system of interest must be partitioned into components. Then, for each component, the appropriate component class must be selected from MESA's *Component Library* or it must be created from scratch with the *Component Tool*

Classes in MESA act as templates from which an infinite number of component instances may be created. Every component class contains a set of specifications for realizing component instances recursively as instances of locales. Additionally, every component class implements a graphical icon that is used to represent component instances in the GUI.

Let us consider a top-down design for the `venturi` component from the EATCS evaporator loop model. The user initiates the design process by selecting the *Component Tool* from the *Workbench* control panel (Figure 6). This action pops up the control panel for the chosen tool (Figure 7). The user must enter a name for the component in the text field. Clicking in the button labeled with a "C" instructs MESA to add the named component class to the component library and then pops up a new workspace for the class.

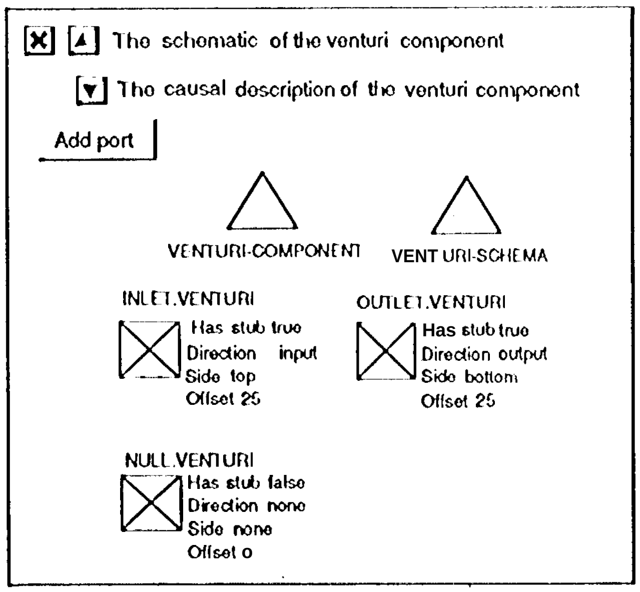


Click on the component icon
and drag its clone to the component library.

Figure 7: The *Component Tool* control panel.

At this point, MESA shifts the user's attention to the workspace associated with the component class. This workspace is used to store the internal representation of classes in MESA. in Figure 8, the two triangle-shaped icons are G2 object definitions (classes). VENTURI - COMPONENT implements the behavior of the component as a set of specifications for quantities and mechanisms. VENTURI - SCHEMA implements a user-customizable icon that is used to graphically display instances of a venturi in causal models. MESA automatically generates the names of these classes from the name entered by the user when the component was created. Initially, only the VENTURI - COMPONENT and VENTURI - SCHEMA icons are present on the venturi component workspace, and the set of specifications in VENTURI - COMPONENT is empty.

Top-down design continues by adding the necessary locale specifications to the venturi component workspace. Locale specifications are


The schematic of the venturi component
The causal description of the venturi component
Add port
VENTURI-COMPONENT
VENTURI-SCHEMA

INLET.VENTURI
Has stub true
Direction input
Side top
Offset 25

OUTLET.VENTURI
Has stub true
Direction output
Side bottom
Offset 25

NULL.VENTURI
Has stub false
Direction none
Side none
Offset 0

Figure 8: The `venturi` component workspace stores class definitions and locale specifications.

implemented as instances of locale-spec classes and represented graphically with a "crossed-square" icon (Figure 8). Again, the user is faced with the choice of selecting a predefined locale from MESA's *Locale Library* or defining one from scratch. The *Locale* Library maybe accessed directly by clicking on the *Add Locale* button on the venturi component workspace. This action pops up a scrolling list from which the user may make a single selection. *Locale Tool* (accessible from the Workbench) provides a locale definition interface that parallels the component definition interface provided by *Component Tool.* A new workspace is created by MESA for each new locale class created by the user. The workspace is initially empty, save for the the AMMONIA-LOCALE and AMMONIA-SCHEMA class icons. For example, Figure 9 shows the workspace associated with the ammonia locale class. Ammonia locales are specified as the inlet and outlet (or source and sink) of the venturi component.

Top-down design proceeds with the addition of quantity specifications to the ammonia locale workspace. However, MESA does not provide a tool for defining new quantity classes. The set of quantity classes (types) is restricted by the set of variable types that EDSE can handle. Accordingly, the user interface for quantity specifications simply enumerates the static set of variable types. The *Add Quantity* button on the ammonia locale

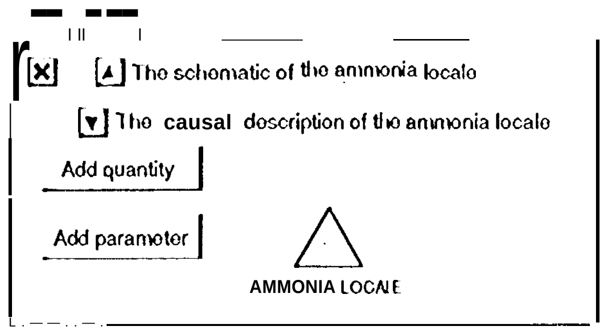
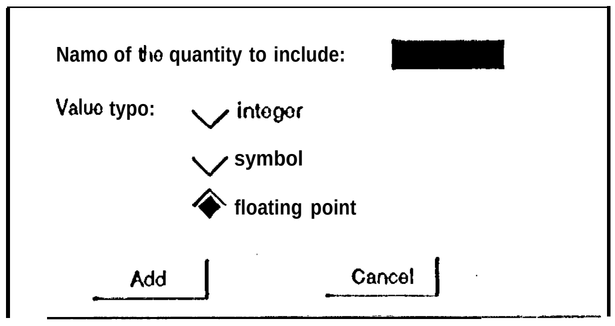

The schematic of the ammonia locale
The causal description of the ammonia locale
Add quantity
Add parameter
AMMONIA LOCALE

Figure 9: The ammonia locale workspace stores quantity specifications.


Name of the quantity to include:

Value type: integer / symbol / [✓ floating point]

Figure 10: Quantities are restricted to three value types-integer, symbol and floating point.

workspace pops up the dialogue shown in Figure 10. The user merely enters a name and chooses the appropriate type to define a quantity specification to be stored on the locale workspace. The *Add Parameter* button pops up the same dialogue; a parameter in MESA is a quantity whose value never changes (i.e., it is a mot node in the causal graph).

Let us move forward in our discussion to the point at which the user has defined all of the locales and quantities necessary to a given component. All that remains to complete the model design for the component is to define a set of mechanisms that mirror the behavior of the physical component at some level of abstraction, The `venturi` component workspace has a subworkspace for describing causal model of the `venturi` component in term.. of primitive quantities and mechanisms (Figure 11). This subworkspace is accessible through a navigation button on the `venturi` component workspace. Quantities arc graphically represented by circular icons with connection stubs on the left and right. Mechanism are represented by similar icons labeled with the letter "M,"

Mechanisms are added to the causal subworkspace by clicking on the *Add Mechanism* button. This action instructs MESA to prompt the user for a mechanism name and to create a new mechanism with that name on the causal subworkspace. The user may then draw with the mouse a link from the right stub of a quantity icon to the left stub of a mechanism icon, thereby defining the quantity as an input to the mechanism. Output quantities are defined by joining the left stub of a quantity icon with the right stub of a mechanism icon.

Mechanism definition is completed by coding the underlying transfer and delay functions of the mechanism. Currently, MESA requires these

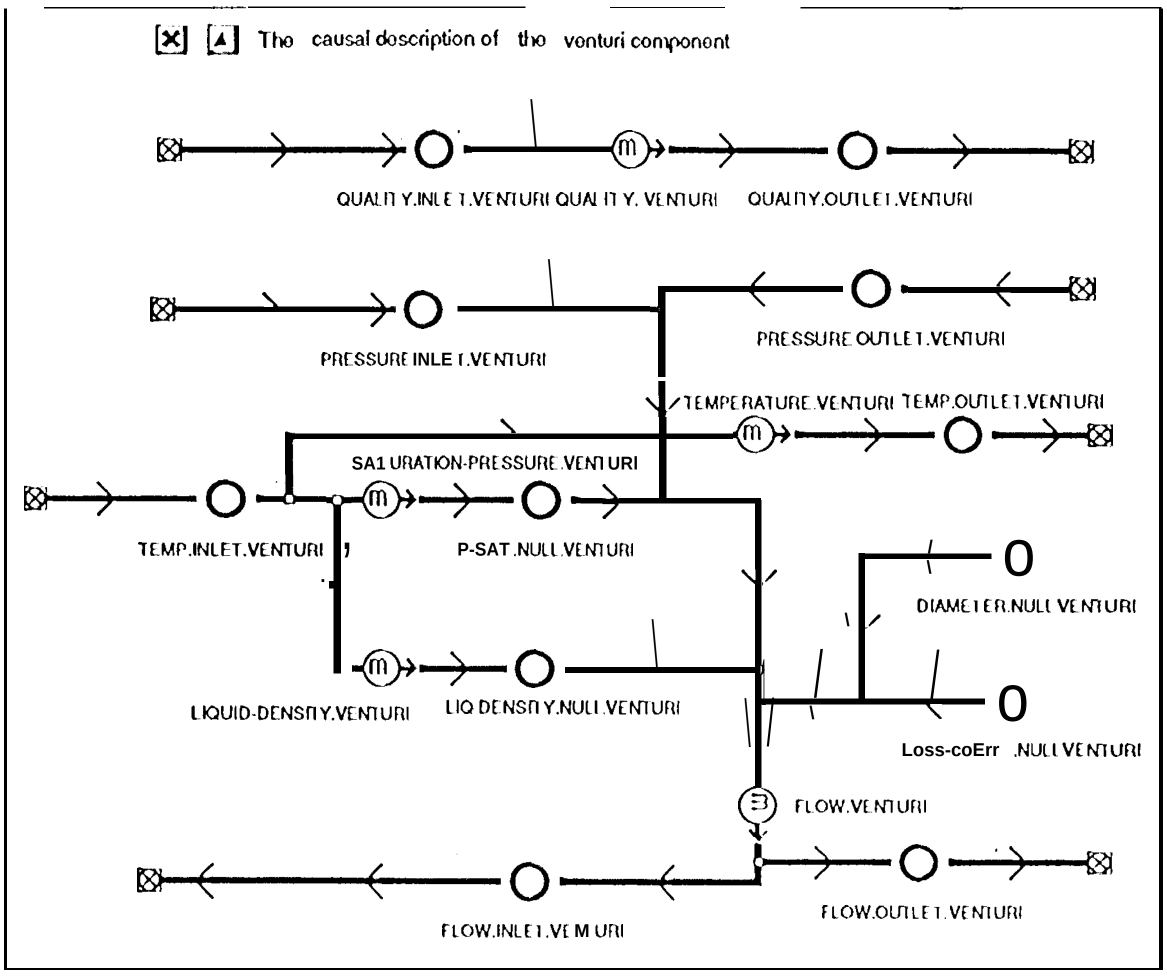


Figure 11: Ix3w-level representation of {he venturi component.

functions to be written as Lisp expressions.[3] The names of the mechanism input quantities may appear as free variable references in a transfer or delay expression. When an expression is evaluated during carnal simulation, all such references are bound (within the lexical scope of the expression) to the current value of the quantity named by the reference. The single value computed by the expression is then propagated to the outputs of the mechanism by the simulator.

3.2.2 Model Assembly

Model assembly is the process of creating instances of components from the appropriate class definition. In contrast to the design process, model assembly is exceedingly simple. Component icons are utilized as a direct-manipulation interface to the model assembly process. icons may be duplicated and moved with the mouse. Connections are established by using the mouse to draw a link between two component icons on a schematic.

Figure 4 shows a simplified schematic of the evaporator loop of the EATCS application. Each component on this schematic was created by instantiating a component from the library using the dialog box accessible

[3] This requirement is actually imposed by the syntax of EDSEL and the implementation of EDSE as a Lisp application program. in future versions of MESA, we hope to provide a graphical method for specifying component behavior.

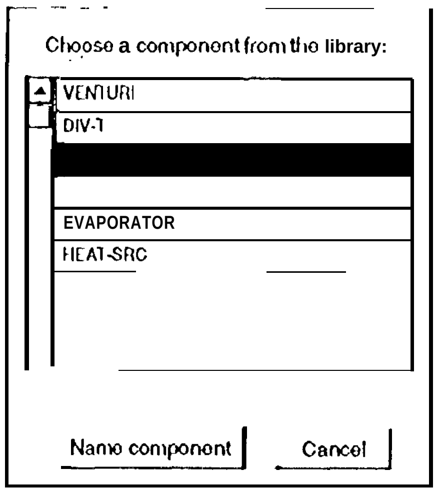


Figure 12: A list of components available in the library.

via the *Add Component* button on the schematic (Figure 12).

4 Status

MESA is an ongoing effort at JPL. The environment described in this report is in use for several projects at JPL and for the EATCS application at McDonnell Douglas Aerospace. The three user processes of MESA are at various stages of development and are discussed separately.

4.1 Model design

Model design involves defining components in terms of locales, quantities and mechanisms. This work occurs entirely within G2 and relies extensively on its user interface. Two factors affect the development of these user interfaces: the flexibility of G2 in its ability to support complex user interactions, and the human factors involved in organizing the locales and components appropriately according to domains, applications, or other criteria.

Human factors are the main driving force in the design of the user interface for model design. With the use of MESA at JPL, and McDonnell Douglas, we expect to gain sufficient experience to provide users with adequate tools for organizing model libraries.

4.2 Model assembly

Model assembly involves two phases: modeling (i.e., selecting component classes from the MESA library according to the component-centered view of the physical system being modeled) and instantiation (i.e., implementing the causal model of the same physical system for model-based reasoning). The latter phase is relatively stable, but there are several open issues in the modeling phase.

Physical system modeling is similar to model design and similar human factor issues also affect the user interface. For example, several versions of physical systems can be modeled. That is, causal models may comprise different component models according to the level of granularity desired by the user. Also, physical systems may be combined in related units by function, location, or other criteria. Finally, several versions of a model may have to be tracked as the design of the components or construction of the physical systems evolves from an initial design.

4.3 Model-based reasoning

The EDSE simulator is the basic model-based reasoning tool used in MESA. The EDSE simulation protocol and the availability of the causal model in EDSE and MESA provide adequate flexibility to implement various model-based reasoning techniques. Two such techniques, sensitivity and cascading-alarm analysis, have already been developed as extensions of the EDSE simulator for the SELMON project. Currently, a diagnosis module and a causal feedback simulation controller are under development for the EATCS application in MESA.

4.3.1 Diagnosis

Part of the original motivation for MESA was to provide a platform to host a domain-independent diagnosis engine for several applications at)] '1, and McDonnell Douglas. For example, diagnosis results can be present ed at the structural (component /connection) level or the behavioral (quantity/mechanism) level according to the user's preference. Furthermore, by integrating several model-based reasoning tools within the same environment, it is possible to experiment with the impact of various combina t ions of such tools. For example, the model developer can try di fferent sensor placements while at the same time stress the diagnosis engine by injecting faults in the model. Sensor placement and fault modeling arc not part of the diagnosis engine; rather, they help in evaluating the diagnosis engine.

4.3.2 Causal feedback

With large physical systems and numerous types of components involved, it is possible that the causal model of a physical system will have feedback loops. Various techniques, such as causal ordering [3] or prediction of initial and final response [4] and behavioral abst raction [6] have been proposed as solutions. in MESA the causal model is automatically analyzed to detect all possible causal loops. MESA presents the results of this analysis to the user, who may then choose a specific feedback analysis technique to be applied,

References

[1] 1.. K. Charest Jr. Building causal models for SELMON: A user's manual for the EDSEL modeling language. Technical report, Artificial Intelligence Group, Jet l'repulsion Laboratory, California, 1992.

[2] R. J. Doyle, S. A. Chien, U. M. Fayyad, and E. J. Wyatt. Focused real-time systems monitoring based on multiple anomaly models. in *Seventh International Workshop on Qualitative Reasoning*, Eastsound, WA, 1993.

[3] Y. Iwasaki and 1. Bhandari. Formal basis for commonsense abstraction of dynamic systems. in *Proceedings of the Seventh National Conference cm Artificial Intelligence*, 1988.

[4] P. Rose and M. A. Kramer. Qualitative analysis of causal feedback. in *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 817-823,1991.

[5] N. F. Rouquette, S. A. Chien, and L. K. Charest Jr. Event-driven simulation in SELMON: An overview of EDSEL. Technical Report 92-23, Artificial intelligence Group, Jet Propulsion Laboratory, California, 1992.

[6] N. F. Rouquette and S. Rajamoney. Dynamic aggregation in qualitat ive physics. in *Model-Based Reasoning Workshop Notes from AAAI-91*. 1991.